

Conceptos de Algoritmos, Datos y Programas 2022

TEORIA 1

TEMAS de la CLASE

- 1 Concepto Informática
- 2 Etapas de resolución de un problema por computadora
- 3 Concepto de Algoritmo
- 4 Concepto de Programa
- 5 Concepto de Dato y Tipo de Dato
- 6 Clasificación de Tipos de Datos Simples definidos por el lenguaje (Entero y Real)
- 7 Datos en Python
- 8 Comenzando un programa en Python
- 9 Ejercitación

Concepto de Informática

La Informática es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.

Ciencia se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas.

La **resolución de problemas** aplicaciones en áreas muy diferentes tales como biología, comercio, control industrial, administración, robótica, educación, arquitectura, etc.

Computadora máquina digital y sincrónica, con capacidad de cálculo numérico y lógico y de comunicación con el mundo exterior. Ayuda al hombre a realizar tareas repetitivas en menor tiempo y con mayor exactitud.

Etapas de resolución de un problema por computadora

Del Problema a la Solución



Cuando se resuelve un problema del mundo real utilizando una computadora es necesario atravesar una serie de etapas.

*Problema del
Mundo Real*



Solución



Etapas de resolución de un problema por computadora

Del Problema a la Solución

Problema
Solución

Problema del
Mundo Real



Análisis

Modelo



1er etapa: se sintetizan los requerimientos del problema y se simplifica el contexto y los datos a utilizar por el programa en la computadora.

Etapas de resolución de un problema por computadora

Del Problema a la Solución

Problema
Solución

Problema del
Mundo Real

Análisis

Diseño

Algoritmos

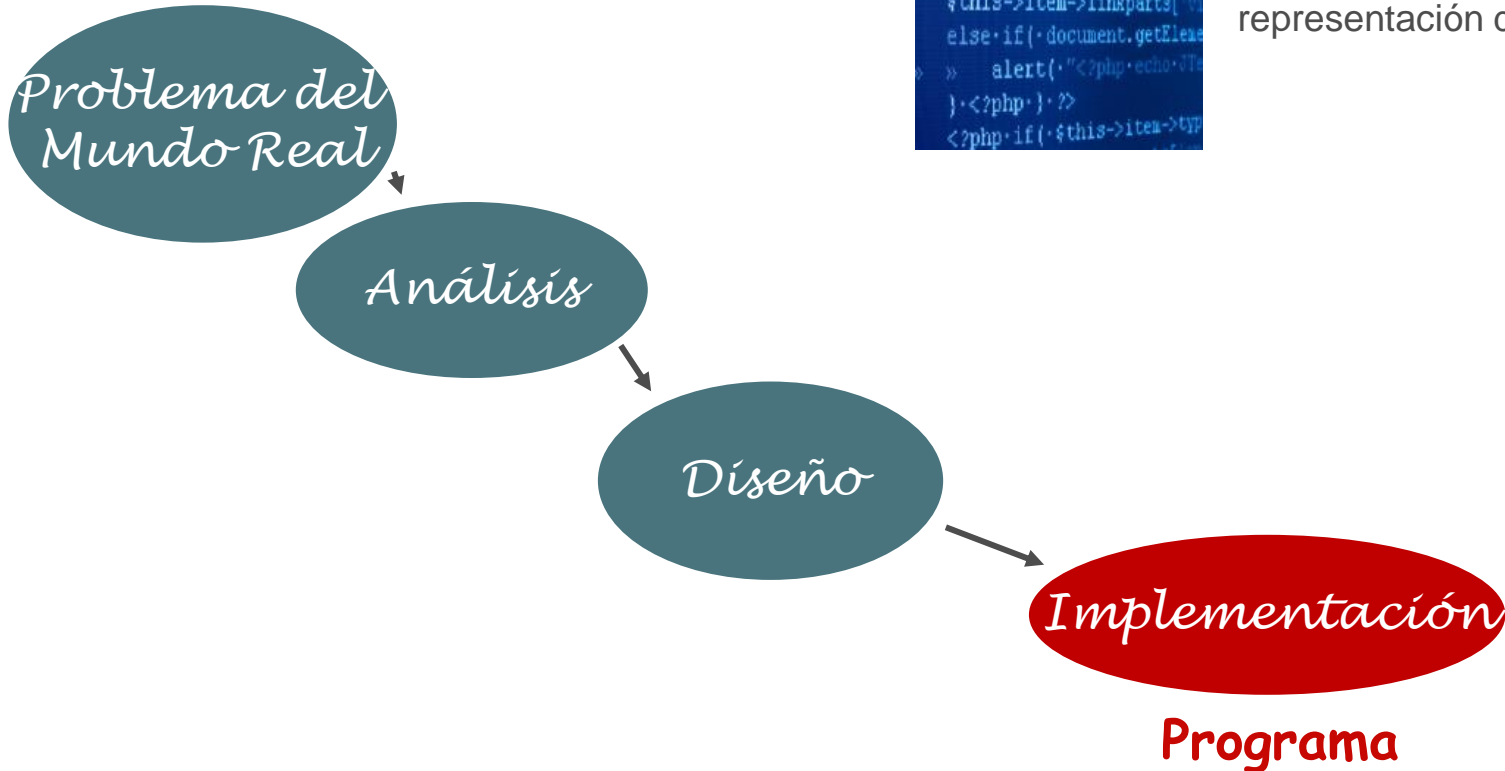
Solución
Modularizada



2da etapa: la descomposición funcional nos ayudará a reducir la complejidad, a distribuir el trabajo y en el futuro a reutilizar los módulos. Algoritmos.

Etapas de resolución de un problema por computadora

Del Problema a la Solución



```
<?php·}·>>
<?php·if(·$this->item->typ
$this->item->linkparts{
$this->item->linkparts{
else·if(·document.getEleas
» alert(·"<?php·echo·$T
}·<?php·}·>>
<?php·if(·$this->item->typ
```

3er etapa: escribir algoritmos en un lenguaje de programación y elegir la representación de los datos.

Etapas de resolución de un problema por computadora

Del Problema a la Solución

Problema
Solución



Etapa de Análisis del Problema

- En una primera etapa, se analiza el problema en su contexto del mundo real. Deben obtenerse los **requerimientos** del usuario. El resultado del análisis del problema es un modelo preciso del ambiente del problema y del objetivo a resolver.

Etapa de Diseño de la solución

- Suponiendo que el problema es computable, a partir del modelo se debe diseñar una solución. En el paradigma procedural, esta etapa involucra entre otras tareas la modularización del problema, considerando la descomposición del mismo y los datos necesarios para cumplir su objetivo.
- Esta etapa involucra la especificación de los algoritmos:
- Cada uno de los módulos del sistema diseñado tiene una función que podemos traducir en un algoritmo (que puede no ser único). La elección del algoritmo adecuado para la función del módulo es muy importante para la eficiencia posterior del sistema de software.

Etapa de Implementación de la solución

- Esta etapa involucra la escritura de los programas. Los algoritmos definidos en la etapa de diseño se convierten en programas escritos en un lenguaje de programación concreto.

Etapa de Verificación de la solución

- Una vez que se tienen los programas escritos y depurados de errores de sintaxis, se debe verificar que su ejecución conduce al resultado deseado, utilizando datos representativos del problema real.

En resumen:

Análisis y Diseño

(NO dependen del lenguaje)

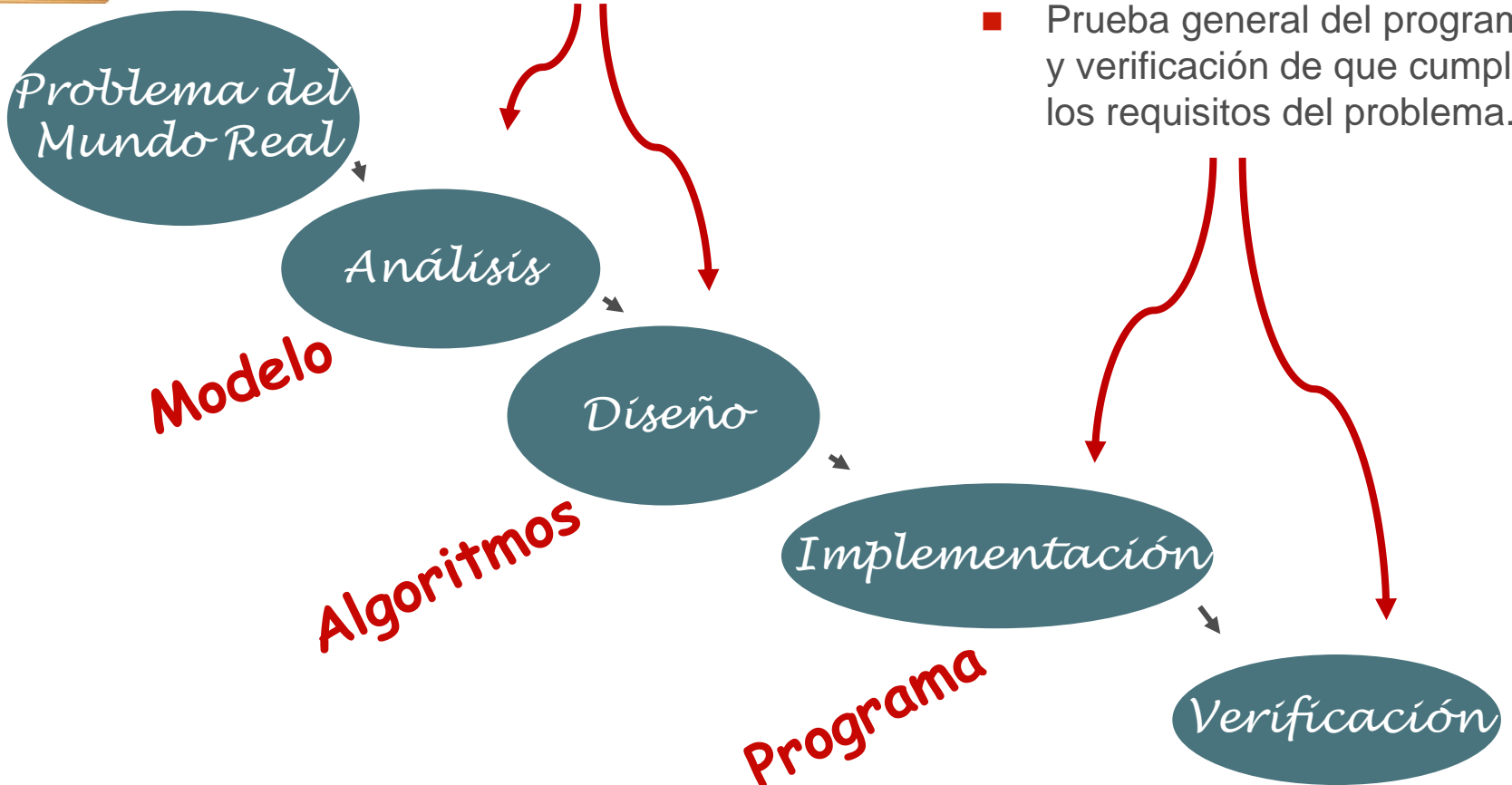
- Entender el problema
- Modelizarlo.
- Modularizar.
- Escribir los algoritmos.

Implementación

(depende del lenguaje)

- Codificación de los algoritmos en un lenguaje de programación.
- Prueba en ejecución de cada módulo.
- Prueba general del programa y verificación de que cumple los requisitos del problema.

Problema
Solución



Concepto de Algoritmo

Definiremos **algoritmo** como la **especificación rigurosa** de la secuencia de pasos (instrucciones) a realizar sobre un **autómata** para alcanzar un resultado deseado en un **tiempo finito**.

✓ *Especificación rigurosa* significa que debemos expresar un algoritmo en forma clara y unívoca.

✓ Alcanzar el resultado en *tiempo finito* significa que suponemos que un algoritmo *comienza y termina*. Por lo tanto, el número de instrucciones debe ser también finito.

Si el autómata es una computadora, tendremos que escribir el algoritmo en un lenguaje "entendible" y ejecutable por la máquina.



PROGRAMA

Concepto de Programa

Definiremos **programa** como el conjunto de instrucciones u órdenes **ejecutables** sobre una **computadora**, que permite cumplir con una función específica (dichas órdenes están expresadas en un lenguaje de programación concreto).

- ✓ Normalmente los programas alcanzan su función objetivo en un **tiempo finito**.
- ✓ Los **programas de aplicación** constituyen el verdadero valor que da utilidad a las computadoras (programas WEB, de administración, cálculo, comunicaciones, control industrial, sistemas expertos etc.).
- ✓ Los **programas** se escriben en un **lenguaje de programación** siguiendo un conjunto de reglas sintácticas y semánticas.

Escribir un programa exige:

- ✓ ***Elegir la representación*** adecuada de los datos del problema.
- ✓ ***Elegir el lenguaje de programación*** a utilizar, según el problema y la máquina a emplear.
- ✓ ***Definir el conjunto de instrucciones*** (en el lenguaje elegido) cuya ejecución ordenada conduce a la solución.

Programa = Instrucciones + Datos

Programación Imperativa

El modelo que siguen los lenguajes de programación para **DEFINIR** y **OPERAR** la información, permite asociarlos a un paradigma de programación particular.

En esta primera parte, trabajaremos bajo el paradigma imperativo/procedural.

Utilizaremos el lenguaje de programación: **Python**



Programa = Instrucciones + Datos



Las **instrucciones** (**acciones**) representan las operaciones que ejecutará la computadora al interpretar el programa.



Se escriben en un lenguaje de programación determinado

Python



Los **datos** son los valores de información de los que se necesita disponer y, en ocasiones transformar, para ejecutar la función del programa.



Cada lenguaje de programación tiene los propios

Concepto de Dato

Un **Dato** es una representación de un objeto del mundo real. Los datos permiten modelizar los aspectos del problema que se quieren resolver mediante un programa ejecutable en una computadora.

Datos constantes: datos que no cambian durante la ejecución del programa.

Datos variables: datos que durante la ejecución del programa pueden cambiar.

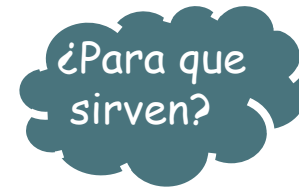
En Python cada dato tiene asociado un tipo de dato, entero, real, lógico, etc

Definición de Tipo de Dato

Un *tipo de dato* es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Cada tipo de dato se caracteriza por presentar:

- ✓ Un rango de valores posibles
- ✓ Un conjunto de operaciones realizables sobre ese tipo
- ✓ Una representación interna



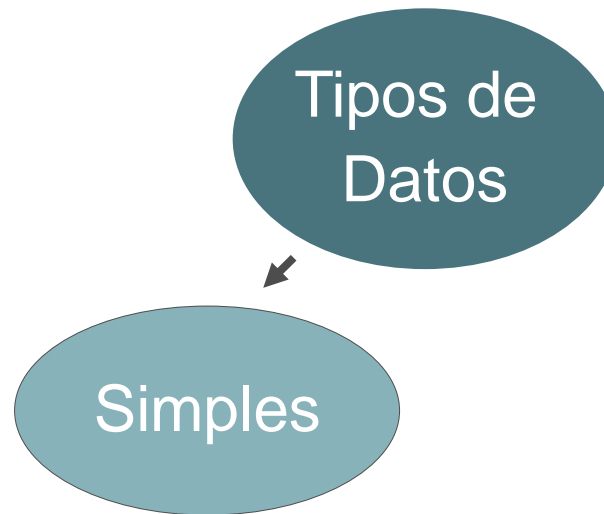
¿Qué valores puedo usar?

¿Qué operaciones puedo aplicar?

¿Cuánta memoria utiliza?

Clasificación de Tipos de Datos

Una primera clasificación

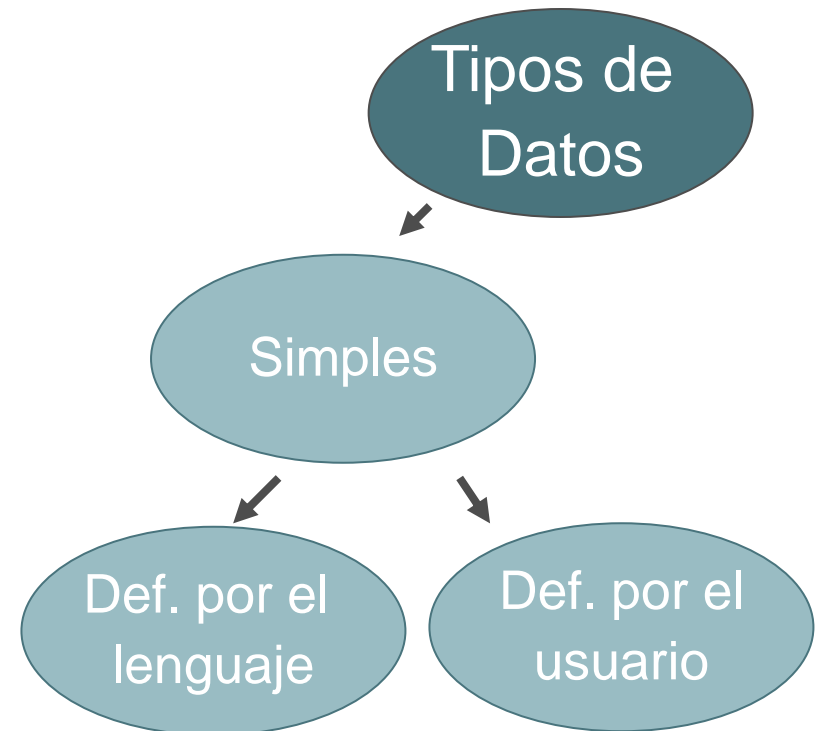


Los tipos de datos simples son aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

Clasificación de Tipos de Datos Simples

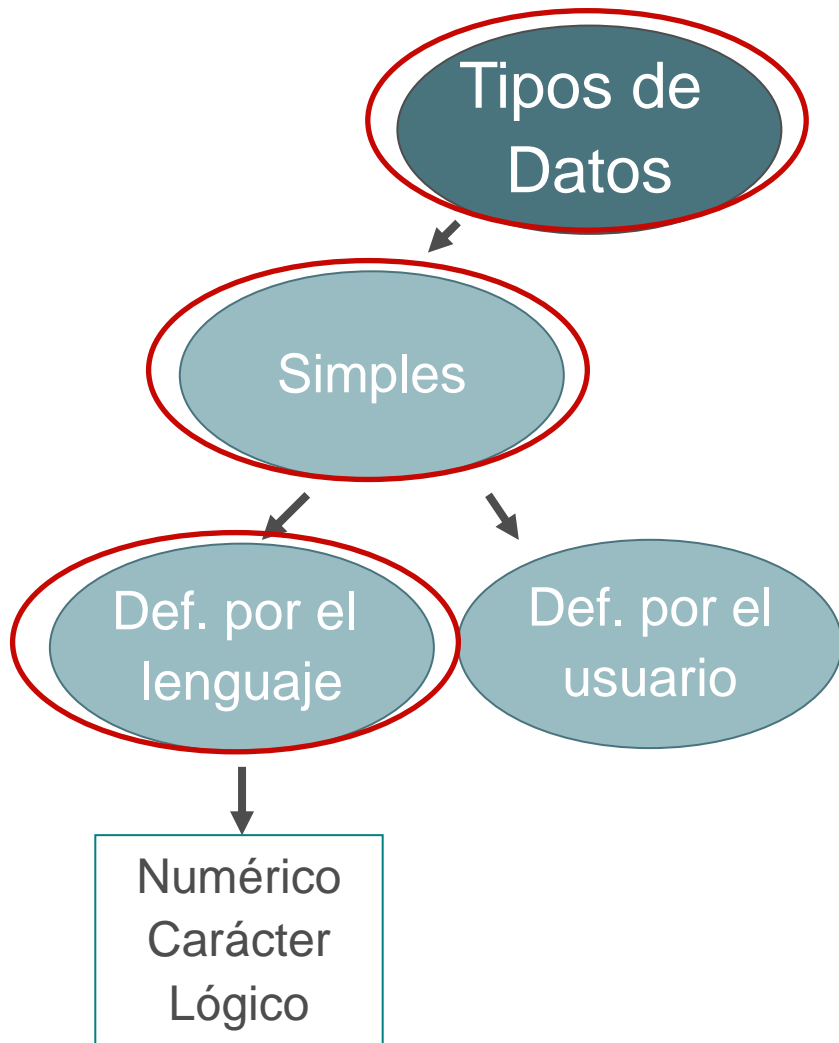
A su vez, los tipos simples, pueden clasificarse en:

- **Tipos de datos definidos por el lenguaje** (primitivos o estándar) son provistos por el lenguaje y tanto la representación como sus operaciones y valores son reservadas al mismo.
- **Tipos definidos por el usuario**, permiten definir nuevos tipos de datos a partir de los tipos primitivos.



Clasificación de Tipos de Datos Simples Definidos por el lenguaje

Comenzaremos presentando los tipos simples y definidos por el lenguaje



Para cada tipo se presentarán:

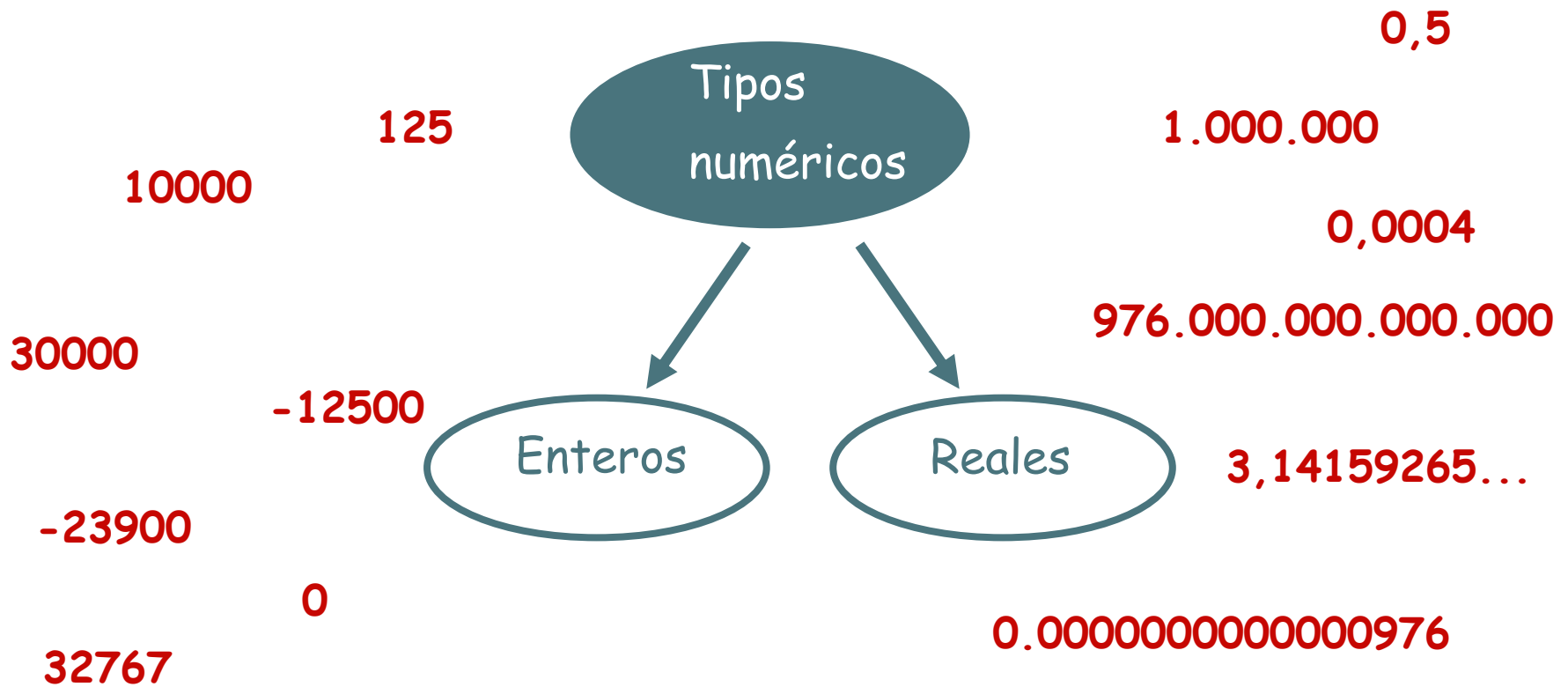
¿Valores posibles?

¿Operaciones?

¿Representación en memoria?

Tipo de dato numérico

El **tipo de dato numérico** permite representar el conjunto de los valores numéricos que puede referirse a los números enteros o a los números reales:



Tipo de Dato numérico: REAL

El **tipo de dato real** es una clase de dato numérico que permite representar números decimales. Es un tipo de dato simple.

- ✓ El tipo de dato real tiene una representación finita de los números reales.
- ✓ Se utiliza la notación exponencial o científica, utilizada por cualquier tipo de calculadora, y consiste en definir cada número como una *mantisa* (parte decimal) y un *exponente* (posición de la coma).

976.000.000.000.000 -> 9,76 × 10¹⁴

0.00000000000000976 -> 9,76 × 10⁻¹⁴

Operaciones del tipo de dato numérico

■ Operaciones aritméticas

Las operaciones válidas para el tipo de dato numérico son:

- ✓ suma (+)
- ✓ resta (-)
- ✓ multiplicación (*)
- ✓ división real (/)
- ✓ división entera (div)
- ✓ módulo (mod)

Ejemplos	Resultado
3+4	7
2,5 / 2	1,25
10 div 3	3
10 mod 3	1 (devuelve el resto de la división)

En general:

Operador	Tipo de Operando	Tipo de resultado
Suma (+)	Entero ó Real	Entero ó Real
Resta (-)	Entero ó Real	Entero ó Real
Multiplicación (*)	Entero ó Real	Entero ó Real
División (/)	Real	Real
Div	Entero	Entero
Mod	Entero	Entero

Orden de precedencia de las operaciones con números

Las expresiones que tienen dos o más operandos requieren reglas matemáticas que permitan determinar el orden de las operaciones.

El orden de precedencia para la resolución de la expresión matemática es:

En primer lugar: *****, **/**, **div**, **mod**

Por último: **+** y **-**

Si existen operaciones con igual orden de precedencia, se resuelven de izquierda a derecha

- $6 + 8 * 5 = 6 + 40 = 46$
- $(6 + 8) * 5 = 14 * 5 = 70$
- $5 * 2 + 7 + 4 * 3 = 10 + 7 + 12 = 17 + 12 = 29$
- $5 * (2 + 7 + 4) * 3 = 5 * 13 * 3 = 195$
- $5 + 10 \text{ mod } 4 = 3$
- $10 \text{ div } 2 * 3 = 15$
- $(5 * 4) \text{ div } 4 = 5$

*Se pueden utilizar
paréntesis para alterar el
orden de precedencia*

Operaciones del tipo de dato numérico

■ Operación de asignación (=)

■ Operaciones de comparación

Además de los operadores matemáticos mencionados, el tipo de dato numérico posee operadores relacionales que permiten comparar valores.

✓ **igualdad** (==),

✓ **desigualdad** (<>)

✓ **orden** (<, <=, >, >=)

▪ $0 < 7$ (verdadero)

▪ $5 <= 7$ (verdadero)

▪ $5 > 7$ (falso)

▪ $5 < > 7$ (verdadero)

▪ $5 >= 2$ (verdadero)

▪ **Pensemos:** $5.0 == 5$???

Su **resultado** es Verdadero o Falso

Tipo de dato lógico

El **tipo de dato lógico** permite representar datos que pueden tomar solamente uno de dos valores. Este tipo de dato se conoce también como tipo de dato boolean. Es un tipo de dato simple y ordinal.

Valores permitidos

➤ **verdadero** (*true*)

➤ **falso** (*false*)

Se utiliza en situaciones donde se representan dos alternativas de una condición.

¿Juan es mayor que María?

¿Edad de Juan?

¿Edad de María?

Tipo de Dato Caracter

El **tipo de dato carácter** es un tipo de dato simple y ordinal.

Un dato de tipo carácter contiene solo un carácter.

Los caracteres que reconocen las computadoras no son estándar. Sin embargo, este conjunto de valores se normalizó, entre otros, por un estándar llamado ASCII, el cual permite establecer un **orden de precedencia** entre los mismos.

Valores permitidos

- ✓ **Caracteres especiales:** '!', '#', '\$', '%', ...
- ✓ **Dígitos:** '0', '1', '2', ..., '8', '9'
- ✓ **Letras mayúsculas:** 'A', 'B', 'C', ..., 'Y', 'Z'
- ✓ **Letras minúsculas:** 'a', 'b', 'c', ..., 'y', 'z'

Tipo de Dato Caracter

Cada elemento del código ASCII tiene asociado un valor entero entre 1 y 255 que permite establecer el orden de precedencia

Caracteres de control ASCII			Caracteres ASCII imprimibles									ASCII extendido											
DEC	HEX	Símbolo ASCII	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	í	193	C1h	ł	225	E1h	ô
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	194	C2h	Ť	226	E2h	Û
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	â	163	A3h	û	195	C3h	Ŧ	227	E3h	Ö
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ñ	196	C4h	ŷ	228	E4h	ö
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	à	165	A5h	Ñ	197	C5h	Ź	229	E5h	Õ
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	â	166	A6h	ª	198	C6h	ž	230	E6h	µ
07	07h	BEL (timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	199	C7h	ž̂	231	E7h	þ
08	08h	BS (retroceso)	40	28h	(72	48h	H	104	68h	h	136	88h	ê	168	A8h	¿	200	C8h	ž̃	232	E8h	þ̂
09	09h	HT (tab horizontal)	41	29h)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	®	201	C9h	ž̄	233	E9h	Û̂
10	0Ah	LF (salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	¬	202	CAh	ž̅	234	EAh	Ü̂
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	½	203	CBh	ž̆	235	EBh	Ü̃
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	î	172	ACH	¼	204	CCh	ž̇	236	ECh	ý
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ï	173	ADh	ı	205	CDh	ž̈	237	EDh	Ÿ
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ā	174	AEnh	«	206	CEh	ž̉	238	EEh	˙
15	0Fh	SI (shift In)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Ā	175	AFh	»	207	CFh	ž̊	239	EFh	˚
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	É	176	B0h	⋮	208	D0h	ž̋	240	F0h	±
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	⋮	209	D1h	ž̌	241	F1h	±̂
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	⋮	210	D2h	ž̍	242	F2h	±̃
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ó	179	B3h	⋮	211	D3h	ž̎	243	F3h	±̄
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ò	180	B4h	⋮	212	D4h	ž̏	244	F4h	±̅
21	15h	NAK (negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ô	181	B5h	⋮	213	D5h	ž̐	245	F5h	±̆
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	ò	182	B6h	⋮	214	D6h	ž̑	246	F6h	±̇
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ù	183	B7h	⋮	215	D7h	ž̒	247	F7h	±̈
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	⋮	216	D8h	ž̓	248	F8h	±̉
25	19h	EM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	Ō	185	B9h	⋮	217	D9h	ž̔	249	F9h	±̊
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	Ū	186	BAh	⋮	218	DAh	ž̕	250	FAh	±̋
27	1Bh	ESC (escape)	59	3Bh	;	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	BBh	⋮	219	DBh	ž̖	251	FBh	±̌
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	⋮	220	DCh	ž̗	252	FCh	±̍
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh]	125	7Dh	}	157	9Dh	Ø	189	BDh	⋮	221	DDh	ž̘	253	FDh	±̎
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	×	190	BEh	⋮	222	DEh	ž̙	254	FEh	±̏
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	_				159	9Fh	f	191	BFh	⋮	223	DFh	ž̚	255	FFh	±̐

elCodigoASCII.com.ar

■ Asignación (=)

■ Comparación (>, <, ==...)

dos valores de tipo carácter se pueden comparar por ==, <>, >, <, >=, <=. El resultado es un valor lógico (verdadero/falso) que depende del orden establecido en el código ASCII

Ejemplos: (ver en la tabla el valor entero asociado a cada caracter)

- ('b' == 'B') *falso*
- ('c' < 'Z') *falso*
- ('c' < 'z') *verdadero*
- ('X' > '5') *verdadero*
- (' ' < 'H') *verdadero*
- ('4' == 4) *no puede evaluarse*
- ('@' > '\$') *verdadero (ya que la ubicación de \$ es 36 y la del @ es 64)*

¿Como se identifican los datos del programa?

IDENTIFICADORES

Para identificar los datos de un programa (constante / variable) se utilizan nombres descriptivos. Esa identificación permite conocer su dirección real en la memoria y el valor que contiene.

En Python, los identificadores están formados por letras, dígitos en cualquier orden y algunos símbolos especiales (excepto el primer carácter que **debe ser una letra**).

flores1

papeles

Total_cuadras

pasos

cantidad

Es obligatorio declarar cada uno de los datos que utiliza el programa

Declaración de constantes y variables en Python

En **Python**, las variables y constantes pueden ser declaradas al principio de un algoritmo, o en cualquier zona del programa, pero por una cuestión de prolijidad siempre es bueno al comienzo.

Ejemplo de constante:

```
CONSTANTE_1 = valor
```

Donde `CONSTANTE_1` es el identificador que representa el nombre de la constante. El tipo de dato de la constante queda definido implícitamente por el valor que se le asigna.

Las constantes van en mayúscula por ejemplo `CONSTANTE_1`, desde el punto de vista funcional no existe ninguna diferencia en Python entre las constantes y variables, pero se recomienda que no se intente modificar el valor de una constante posteriormente, solo como buena práctica.

En el caso de las variables

Una variable, es un espacio en la memoria del PC al que se le da un nombre para almacenar un valor con un tipo específico.

En Python la variable se declara asignándole un valor, en otras palabras no es necesario declararlas previamente, con solo asignarle algún valor se creará en el momento:

```
variable = valor
```

Por ejemplo:

```
variable_1 = "U.T.N."  
  
x = 5  
  
caso_22 = 89
```

Cambiar el valor o el tipo de datos en las variables

Es muy fácil, para cambiar el valor de una variable lo hacemos simplemente con el operador de asignación "=" volvemos a agregarle (asignarle) un nuevo valor, si queremos ingresar texto en una variable que tenía valor numérico no es necesario hacer más nada solo ingresar el nuevo valor veamos:

Por ejemplo:

```
variable_1 = "U.T.N." (Es texto)

variable_1 = 54 (Numérico)

variable_1 = 89 (cambio su valor)
```

Tipos de datos en Python

En Python, para representar los tipos de datos vistos se cuenta con:

- **int** ➡ dato numérico entero
- **float** ➡ dato numérico real
- **boolean** ➡ dato lógico
- “ “ ➡ dato carácter

Cuando se asigna el tipo de dato a una variable queda determinado:

Valores posibles / Operaciones permitidas / Representación en memoria

Operación de entrada en Python: Input

■ Input

Se usa para leer datos (por defecto desde teclado) y asignarlos a las variables correspondientes.

```
nombre = input("Ingrese su nombre")
```

```
valor_1= int(input (a))
```

```
valor_2 = float(input(b))
```

Operación de salida en Python: Print

■ Print

se usa para mostrar el contenido de una variable, por defecto en pantalla. Pueden ser de tipo entero, real, char. Los datos a mostrar si son más de uno deben ir separados por coma.

```
print (a)
```

```
print ("El valor es: ", a)
```

```
print (f"El valor es {b}")
```

*** Este print usa la función "format"*

Operación de salida en Python: Print

■ Mismo resultado – diferentes formas de escribir

```
@author: Fabian
```

```
"""
```

```
# Pide nombre y edad, luego saluda
```

```
# Forma Clásica
```

```
nombre=input("Ingresa un nombre: ")
```

```
edad=int(input("Cual es tu edad?: "))
```

```
print ("Hola",nombre,"como estas?","veo que  
tienes",edad, "años")
```


Operación de salida en Python: Print

■ Mismo resultado – diferentes formas de escribir

```
@author: Fabian
```

```
"""
```

Forma más usada

```
nombre=input("Ingrese un nombre: ")
```

```
edad=int(input("Cual es tu edad?: "))
```

```
print ("Hola {} como estas? veo que tienes {}  
años".format(nombre,edad))
```

Operación de salida en Python: Print

■ Mismo resultado – diferentes formas de escribir

```
@author: Fabian
```

```
"""
```

```
# Forma más simple
```

```
nombre=input("Ingresa un nombre: ")
```

```
edad=int(input("Cual es tu edad?: "))
```

```
print (f"Hola {nombre}, como estas? veo que  
tienes {edad} años")
```

Operadores aritméticos en Python

Prioridad de los Operadores Aritméticos	
1	Paréntesis <code>()</code>
2	Exponenciación <code>**</code>
3	Multiplicación, División y Módulo <code>*</code> , <code>/</code> , <code>%</code>
4	Suma y Resta <code>+</code> , <code>-</code>

Operadores Lógicos en Python

And (Conjunción)	and
Or (Disyunción)	or
Negación	not

Operadores Relacionales en Python

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
!=	Diferente
==	Igual

Practiquemos los conceptos!!



Gracias

